Final Exam - 10-405

May 2, 2018

10-405 **Name:**_____
Spring 2018
Final Exam Answer Key **Andrew ID:**_____
Time Limit: 80 Minutes

Grade Table (for teacher use only)

Question	Points	Score
1	12	
2	12	
3	6	
4	14	
5	10	
6	16	
7	6	
8	4	
Total:	80	

1. (12 points) TA: Vivek Shankar **Deep Learning**
Consider the following architecture for the neural network shown below.

$$z_1 = XW_1 + b_1$$
$$O^1 = \sigma(z_1)$$
$$z_2 = XW_2 + b_2$$
$$O^2 = \sigma(z_2)$$

Recall matrix multiplication: $M = A \cdot B$ where A is $n \times m$, and B is $m \times k$. Then, both M and the backpropagated derivative of the loss w.r.t. M ($\frac{\partial \text{loss}}{\partial M}$) will be $n \times k$. To get the partial derivatives of the loss w.r.t. A and B:

$$\frac{\partial \text{loss}}{\partial A} = \frac{\partial \text{loss}}{\partial M} \cdot B^T$$
$$\frac{\partial \text{loss}}{\partial B} = A^T \cdot \frac{\partial \text{loss}}{\partial M}$$

. Here, $x \in \mathbb{R}^{1\times n}$ is the layer input. $W_1, W_2 \in \mathbb{R}^{n\times n}$. $b_1, b_2 \in \mathbb{R}^{1\times n}$. The non-linearity σ (sigmoid) is applied element-wise.

(a) (6 points) Using the chain rule, derive the partial derivative of O^2 w.r.t. W_1, i.e., $\frac{\partial O^2}{\partial W_1}$. You can leave your answer in terms of any of the variables above. Also, compute $\frac{\partial O^2}{\partial z_2}$, $\frac{\partial O^2}{\partial O^1}$, and $\frac{\partial O^2}{\partial z_1}$; these partial derivatives will be helpful in determining $\frac{\partial O^2}{\partial W_1}$. Finally, provide the dimensions of the result of each partial derivative below. We have provided the result of the first partial derivative for you below.

$\frac{\partial O^2}{\partial z_2}$ = ____$\sigma'(z_2)$_____, Dimension: ____$\mathbb{R}^{1\times n}$_____

$\frac{\partial O^2}{\partial O^1}$ = _____, Dimension: _____

$\frac{\partial O^2}{\partial z_1}$ = _____, Dimension: _____

$\frac{\partial O^2}{\partial W_1}$ = _____, Dimension: _____

$\frac{\partial O^2}{\partial z_2} = \sigma'(z_2) \in \mathbb{R}^{1\times n}$
$\frac{\partial O^2}{\partial O^1} = \frac{\partial O^2}{\partial z_2} W_2^T \in \mathbb{R}^{1\times n}$
$\frac{\partial O^2}{\partial z_1} = \frac{\partial O^2}{\partial O^1} * \sigma'(z_1) \in \mathbb{R}^{1\times n}$ ($*$ is element-wise multiplication)
$\frac{\partial O^2}{\partial W_1} = X^T \frac{\partial O^2}{\partial z_1} \in \mathbb{R}^{n\times n}$
Putting it all together, we get:
$\frac{\partial O^2}{\partial W_1} = X^T(\sigma'(z_2)W_2^T * \sigma'(z_1)) \in \mathbb{R}^{n\times n}$

(b) (2 points) Imagine that the structure of the above network were repeated n times, for large n. This network suffers from the vanishing gradient problem. Which of the following strategies will help to address the vanishing gradients problem? Circle all that apply.

A. Adding more layers to the neural network.
B. Removing layers from the neural network.
C. Using the sigmoid activation function.
D. Replacing the sigmoid with an ReLu activation function.
E. Adding L2 regularization

(c) (2 points) In Assignments 4 and 5, we saw that we could use gradient check to check the accuracy of our analytically computed gradients. Rather than analytically computing the gradient as we did above, why don't we use the numerical gradients to implement backpropagation and train neural networks in practice? (Give one plausible reason in 1-2 sentences.)

It becomes too expensive to run gradient check to compute gradients for backpropagation in practice. Doing so requires iterating through all values in each of the parameters and computing two forward passes for each iteration. Also, numerical gradients will be less accurate.

(d) (2 points) It is always a good idea to initialize all the weights (including bias terms) of a neural network to zeros. Circle either True / False.

False. All the neurons will receive the same gradient update if they all start off with the same initialization.

2. (12 points) TA: Vivek Shankar **Stochastic Gradient Descent**

(a) **(6 points)** Below is a partially completed implementation of lazy L2 regularized stochastic gradient descent with sparse updates. Fill in the blanks in the code below.

```
A = HashTable() //stores last decay update times
B = HashTable() //stores parameter weights
k = 0
for t = 1...T:
    k = k + 1
    for each example (x_i, y_i):
        p = ... //prediction for x_i
        for each non-zero feature x_i^j with index j:
            if j not in B: B[j]=0
            if j not in A: A[j]=0
            B[j] = B[j](1 - 2λμ)^(_____________)

            B[j] = B[j] + _____________

            A[j] = _____________
```

Answers:
(1) $B[j] = B[j](1 - 2\lambda\mu)^{(k-A[j])}$
(2) $B[j] = B[j] + \lambda(y_i - p)x_i^j$
(3) $A[j] = k$
In the true/false questions below, circle the correct answers.

(b) **(2 points)** In unregularized stochastic gradient descent, we only need to loop over the **non-zero** features of each example x_i because the gradient of w^j is zero if $x_i^j = 0$

A. True
B. False

(c) **(2 points)** The hash trick for lazy, regularized, stochastic gradient descent with sparse updates becomes ineffective when there are collisions between words.

A. True
B. False

(d) **(2 points)** In lazy, regularized stochastic gradient descent as we descussed in class, we need to apply the weight decay to features in the

example **before** we compute the prediction (note: there may be a bug in the implementation above!)

A. True

B. False

3. (6 points) **Latent Dirichlet Allocation and Spark**

In the equations below, r s and t are are summations of values for each topic t, $n_{t|d}$ is the number of times topic t has been assigned in document d, $n_{w|t}$ is the number of times word w has been assigned to topic t, and $n_{\cdot|t} = \sum_w n_{w|t}$. Assume $\forall t, \alpha_t > 0$ and that $\beta > 0$.

$$s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}}$$

$$r = \sum_t \frac{n_{t|d} \beta}{\beta V + n_{\cdot|t}}$$

$$q = \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{\cdot|t}}.$$

Let F_w be the frequency of word w in the corpus. In which summation are at most F_w terms in the sum non-zero ? **s** / **r** / **q**

In which summation are all the terms in the sum non-zero? **s** / **r** / **q**

Let N_d be the length of the document. In which summation are at most N_d terms non-zero? **s** / **r** / **q**

4. (14 points) TA: Sarthak Garg **Randomized Algorithms**

(a) **(2 points)** Suppose you have 2 Bloom filters B_1 and B_2, and their hash functions are independent. You add the same set S of m items to both these Bloom filters. After adding these items, B_1 and B_2 give the wrong answer to the set membership query with probabilities at most p_1 and p_2 respectively. Design a test for membership in S using both these bloom filters, which has an error probability $< \min(p_1, p_2)$. Describe your test in 2-3 lines:

We can query for membership in both the bloom filters. If any one of them outputs False, then we can be assured that the answer is False. Thus an error will be made iff both the filters make an error. The probability of that is less than $p_1 p_2$

(b) **(2 points)** For your test, what is a tight upper bound on the probability of an error?

$p_1 p_2$

(c) **(4 points)** Your friend gives you a pre-computed Bloom filter in which 60% of the bits are set. Your friend claims that the filter gives the correct answer to a membership query with probability at least 0.64. How many hashers must the filter be using? (Circle the correct answer).

A. 1

B. 2

C. 3

D. 4 or more

Explain your answer in 1-2 sentences.

Assume that the filter has k hashers. Consider an element that does not belong to the filter. Each of the k hashes of this element will land on a set bit with probability $\frac{3}{5}$. This implies that all the k hashes of this element will be set with probability $(\frac{3}{5})^k$. Thus we have $(\frac{3}{5})^k = 1 - 0.64$. Solving for k gives $k = 2$

(d) **(4 points)** Recall that a countmin sketch supports two operations, which are implemented as follows for a sketch with K rows:

Add(A, x, y):

```
for k = 1...K:
    A[k][hash(k + x)] += y
```

Get(A, x)

```
m = inf
for k = 1...K:
    m = min(m, A[k][hash(k + x)])
return m
```

You are given two corpora (multisets of words) C_1 and C_2, and your goal is to compute the difference in word frequencies: i.e., for each word w, compute $freq(w, C_1) - freq(w, C_2)$. Consider these two strategies.

A. Use a single count-min sketch A. For every word w in C_1, do $Add(A, w, 1)$. Then for every word w in C_2, do $Add(A, w, -1)$. For estimating the frequency difference for word x, return $Get(A, x)$.

B. Use two count min sketches A and B. For every word w in W_1, do $Add(A, w, 1)$. For every word w in W_2 do $Add(B, w, 1)$. For estimating the frequency difference for word x, compute $Get(A, x) - Get(B, x)$

Which of the two alternatives is preferable? Describe why in at most 1-2 sentences.

Only version B is correct. We cannot add negative values into a count-min sketch, without losing the guarantees on errors.

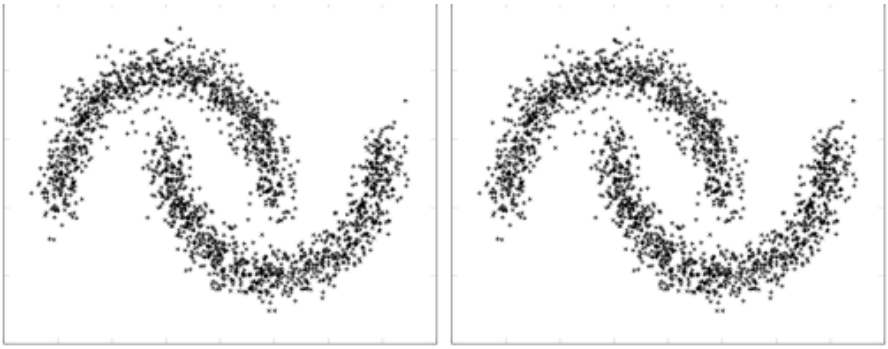
(e) **2 points** True or false: In locality sensitive hashing (LHS), the signature of $\mathbf{x}$ is a vector $\mathbf{bx} \in \mathbb{R}^k$ where k is called the *signature length*. **True** / **False**

5. (10 points) TA: Sarthak Garg **Unsupervised Learning on Graphs.**

   (a) **4 points** Mark the questions below as True or False

   - Spectral clustering can be performed in time linear in the number of edges. **True**/**False**
     Spectral Clustering is not scalable because of eigenvector computation is a bottleneck (ref. from slides)
   - For two clusters, spectral clustering solution is exactly equivalent assigning vertices to discrete classes with a vector $\mathbf{y}$ of +1,-1 values, such that the NCUT criteria $\mathbf{y}^T(D-W)\mathbf{y}$ is minimized. **True**/**False**
     Spectral clustering is equivalent to 'soft' version of the NCUT problem, where vertices are not assigned discrete classes like +-1

   (b) **2 points** Below are two copies of a 2D dataset. On the left indicate clusters that might be found by spectral clustering. On the right indicate clusters that might be found by k-Means.



Spectral K Means

   (c) **4 points** Consider adapting PIC to document clustering. Let N and D be the diagonal matrices used for normalization and the matrix F is defined so that $F(i,k)$ is the TFIDF weight of word w_k in document v_i. Which of the following ordering is preferable? Circle it and **explain why** in 1-2 sentences.

$$v^t = Wv^{t-1} = D^{-1}Av^{t-1} = D^{-1}(N^{-1}(F^T(F(N^{-1}v^{t-1})))) \quad (1)$$

$$v^t = Wv^{t-1} = D^{-1}Av^{t-1} = ((((D^{-1}N^{-1})F^T)F)N^{-1})v^{t-1} \quad (2)$$

The order of updates for the first equation (1) is more efficient as it involves only matrix x vector operations

6. (16 points) TA: Nitish **Graph-based Architectures**

(a) **(12 points)** Among well-known spatial clustering algorithms, `DBSCAN` is an effective algorithm popular for identifying arbitrary number of clusters in the data based on spatial density and identifying outliers. Let us implement a simple parallelized version of this algorithm.

Consider a graph where two nodes are connected by an edge *iff* the distance between them is less than ϵ. A node is a *core* node *iff* it is connected to at least N nodes. The objective of the `DBSCAN` algorithm is to assign each node its corresponding cluster id such that, upon convergence,

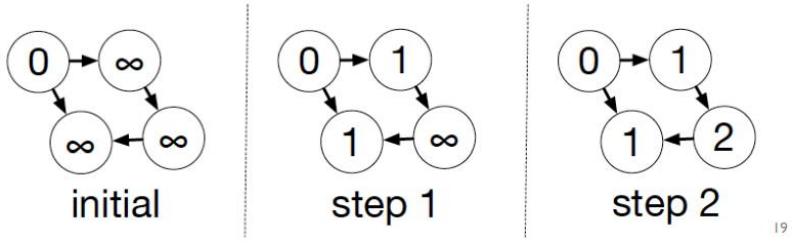
- Two adjacent *core* nodes must have the same cluster id.
- A *non-core* node must have the same cluster id as at least one of the core nodes it is connected to. If it is not connected to any *core* node, it is an *outlier*, and need not be assigned to any cluster.

You can assume that:

- the state of each node, upon convergence, is (*node id*, *cluster id*), where *node id* is the unique id of the node, and *cluster id* is the id of the cluster the node belongs to
- `getId()` returns a unique id (e.g., "3") for each node (*node id*)
- `nEdges(source)` returns the number of nodes connected to `source`
- the *cluster id* for each cluster is the smallest value among the *node id*s of the *core* nodes in the cluster
- the *cluster id* of an outlier (node that is not connected to any *core* node) is `infinity`

Below is a particular parallelized implementation of this algorithm using signal/collect. Fill in the missing parts. As a hint, here is the code for single-source shortest path:

initialState	`if (isSource) 0 else infinity`
collect()	`return min(oldState, min(signals))`
signal()	`return source.state + edge.weight`



```
def initialState:

    (_____________________________, infinity)


def collect():

    if _____________________________:

        return (oldState[0], _____________________________)

    else:

        return _____________________________

def signal():

    if _____________________________:

        return _____________________________

    else:
        return infinity
```

Answer:

```
initialState:
    (getid(), infinity)
collect():
    if len(signals)>0:
        return (oldState[0], min(oldState[1], min(signals)))
    else:
        return oldState
signal():
    if nEdges(source) >= N:
        return source.state[1]
    else:
        return infinity
```

End Answer

(b) **(2 points)** Bob intends to execute a graph-based algorithm on a large graph with billions of edges. Since he does not have access to any distributed computing resources, he decides to use just his personal machine for the execution. Which of the following systems/architectures would be *most* appropriate for him to use? (Choose a single answer.)

A. GraphChi
B. Pregel
C. PowerGraph
D. GraphX

(c) **(2 points)** For each of the algorithms below, circle the framework/system that that is *most* natural for parallelism (circle exactly one choice):

PageRank	Pregel,	MapReduce,	Either
SGD for Deep Neural Networks	GraphLab,	GPUs,	Either

Answers: Pregel, GPUs

7. (6 points) TA: Vidhan **Parameter Servers**

   (a) **4 points** Mark True or False

   - A completely asynchronous execution of a parallelisable ML algorithm often leads to much faster convergence than a bounded asynchronous one **True** / **False**
   - In the SSP (Stale Synchronous Parallel) model, if we set staleness = 0, it will be similar to a fully asynchronous parameter server **True** / **False**
   - Collapsed Gibbs Sampling for LDA is a good use case for the parameter server architecture **True** / **False**
   - In a text processing task, sorting the words by frequency and distributing the corresponding weights across servers such that one server gets the least frequent N/P words and another gets the most frequent N/P words is an effective partitioning strategy **True**/**False**

   (b) **2 points** In 1-2 sentences, state one disadvantage of each setting.

   - Keeping staleness = 0
   - Keeping staleness = very high number

   Staleness = 0 : Each worker needs to wait a lot. Stragglers slow down every worker.

   Staleness = infinity. The parameters may become too stale, and may diverge completely, leading to a non-optimal convergence.

8. (4 points) TA: Nitish **SSL on Graphs**

   (a) **2 points** Select which of the following is **not** true about MAD (Modified Adsorption) algorithm.

   A. A countMin sketch can be used to effectively scale MAD to many labels

   B. It involves a hard assignment of the seed labels

   C. It involves a prior over the distribution of seed labels

   D. Both A & C

   Answer : B

   (b) **2 points** True or false: The seed labels are reset in the MultiRankWalk method but not in the Harmonic Field method. **True** / **False**